#include "U8glib.h"

U8GLIB\_SSD1306\_128X64 u8g(U8G\_I2C\_OPT\_DEV\_0 | U8G\_I2C\_OPT\_NO\_ACK | U8G\_I2C\_OPT\_FAST); // Fast I2C / TWI

// U8GLIB\_SSD1306\_128X64 u8g(13, 11, 8, 9, 10); // SPI connection

// GND > GND

// VCC > 5V

// SCL > 13

// SDA > 11

// RES > 10

// DC > 9

// CS > 8

// 'R Gauge', 128x64px

const unsigned char epd\_bitmap\_R\_Gauge [] PROGMEM = {

0x7f, 0x80, 0x00, 0x07, 0xfe, 0x00, 0x01, 0xc7, 0xf8, 0x00, 0x03, 0x9f, 0xf8, 0x1f, 0xf1, 0xfe,

0xff, 0xc0, 0x00, 0x07, 0xfe, 0x00, 0x01, 0xef, 0xfc, 0x00, 0x03, 0xdf, 0xf8, 0x1f, 0xfb, 0xff,

0xc0, 0xc0, 0x00, 0x06, 0x00, 0x00, 0x00, 0x6c, 0x0c, 0x00, 0x00, 0xd8, 0x00, 0x00, 0x1b, 0x03,

0xc0, 0xc0, 0x00, 0x07, 0xfc, 0x00, 0x00, 0x6c, 0x0c, 0x00, 0x00, 0xdf, 0xf0, 0x0f, 0xfb, 0x03,

0xc0, 0xc0, 0x00, 0x07, 0xfe, 0x00, 0x00, 0x6c, 0x0c, 0x00, 0x00, 0xdf, 0xf8, 0x1f, 0xf3, 0x03,

0xc0, 0xc0, 0x00, 0x00, 0x06, 0x00, 0x00, 0x6c, 0x0c, 0x00, 0x00, 0xc0, 0x18, 0x18, 0x03, 0x03,

0xff, 0xc0, 0x00, 0x07, 0xfe, 0x00, 0x00, 0x6f, 0xfc, 0x00, 0x00, 0xdf, 0xf8, 0x1f, 0xfb, 0xff,

0x7f, 0x80, 0x00, 0x07, 0xfc, 0x00, 0x00, 0x67, 0xf8, 0x00, 0x00, 0xdf, 0xf0, 0x1f, 0xf9, 0xfe,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x20, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x80, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x04,

0x20, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x80, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x04,

0x20, 0x00, 0x20, 0x00, 0x40, 0x00, 0x40, 0x00, 0x80, 0x00, 0x80, 0x01, 0x00, 0x02, 0x00, 0x04,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xfd,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xfd,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xfd,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfd,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfd,

0x80, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfd,

0x80, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfd,

0xbf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfd,

0xbf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfd,

0xbf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfd,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,

0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x01, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x01, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x01, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x03, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x03, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x03, 0xe0, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x03, 0xe0, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x07, 0xc0, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x07, 0xc0, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x07, 0xc0, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x07, 0xc0, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x0f, 0x87, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x0f, 0x8f, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x0f, 0x8f, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x0f, 0x8f, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x1f, 0x03, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x1f, 0x01, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x1f, 0x00, 0xff, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x1f, 0x00, 0x3f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x3e, 0x00, 0x1f, 0xf0, 0x00, 0x7f, 0xc7, 0xf8, 0xff, 0x1f, 0xf7, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x3e, 0x00, 0x07, 0xf8, 0x00, 0x7f, 0xef, 0xfd, 0xff, 0xbf, 0xf7, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x3e, 0x00, 0x03, 0xfe, 0x00, 0x60, 0x6c, 0x0d, 0x81, 0xb0, 0x00, 0x60, 0x00, 0x00, 0x00,

0x00, 0x3c, 0x00, 0x00, 0xff, 0x00, 0x7f, 0xec, 0x0d, 0x81, 0xbf, 0xe0, 0x60, 0x00, 0x00, 0x00,

0x00, 0x7c, 0x00, 0x00, 0x7f, 0xc0, 0x7f, 0xec, 0x0d, 0x81, 0x9f, 0xf0, 0x60, 0x00, 0x00, 0x00,

0x00, 0x7c, 0x00, 0x00, 0x1f, 0xe0, 0x60, 0x6c, 0x0d, 0x81, 0x80, 0x30, 0x60, 0x00, 0x00, 0x00,

0x00, 0x7c, 0x00, 0x00, 0x0f, 0xf8, 0x7f, 0xef, 0xfd, 0xff, 0xbf, 0xf0, 0x60, 0x00, 0x00, 0x00,

0x00, 0x78, 0x00, 0x00, 0x07, 0xf8, 0x7f, 0xc7, 0xf8, 0xff, 0x3f, 0xe0, 0x60, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// Array of all bitmaps for convenience. (Total bytes used to store images in PROGMEM = 1040)

const int epd\_bitmap\_allArray\_LEN = 1;

const unsigned char\* epd\_bitmap\_allArray[1] = {

epd\_bitmap\_R\_Gauge

};

int progress = 0; // progress of the gauge

int frame = 0; // frame for the inner part of the icon

void setup() {

u8g.setFont(u8g\_font\_tpssb);

u8g.setColorIndex(1);

pinMode(A0, INPUT); // set the pinmode for A0 to input to read the MAP sensor value

}

// conversion and calibration for Bosch 0281002593 MAP Sensor

//#include <Wire.h>

//#define sensorPin A0

//float rawval = 0; // Setup raw sensor value

//float barboost = 0; // Setup value for boost bar

//void setup()

//{

//Serial.begin(9600); // start monitoring raw voltage for calibration

//for (count = 0; count <= 128; count++) //zero all elements

//{

//sensorArray[count] = 0;

//}

//}

//void loop() // Start loop

//{

//int boostmbar = map(analogRead(sensorPin), 21, 961, 100, 2600);

//rawval = analogRead(0); // Read MAP sensor raw value on analog port 0

//barboost = (((rawval \* 0.19) - 69.45) + 10); // Calculate boost value

//if ((((boostmbar \* 0.001) - 0.865)\*14) < 0) //use for displaying vacuum

//do { (boostmbar \* 0.001) - 0.865)\*63.2)

//else if ((((boostmbar \* 0.001) - 0.865)\*14) > 0) // use for displaying boost

void loop() {

u8g.firstPage();

do {

u8g.setColorIndex(1);

u8g.drawBitmapP( 0, 0, 128/8, 64, epd\_bitmap\_R\_Gauge); // fullscreen image

u8g.setColorIndex(0); //black color

u8g.drawBox(1, 16, 126, 12);

u8g.setColorIndex(1); // progress bars

u8g.drawBox(2, 24, map(progress, 0, 100 , 0, 17), 3); // bar for 0:2.5

u8g.drawBox(19, 23, map(progress, 0, 100 , 0, 15), 4); // bar for 2.5:5

u8g.drawBox(34, 22, map(progress, 0, 100 , 0, 16), 5); // bar for 5:7.5

u8g.drawBox(50, 21, map(progress, 0, 100 , 0, 15), 6); // bar for 7.5:10

u8g.drawBox(65, 20, map(progress, 0, 100 , 0, 17), 7); // bar for 10:12.5

u8g.drawBox(81, 19, map(progress, 0, 100 , 0, 16), 8); // bar for 12.5:15

u8g.drawBox(97, 18, map(progress, 0, 100 , 0, 14), 9); // bar for 15:17.5

u8g.drawBox(111,17, map(progress, 0, 100 , 0, 15), 10); // bar for 17.5:20

} while ( u8g.nextPage() );

if (progress < 100) {

progress++;

} else {

progress = 0;

//progress = analogRead(A0);

}

}